Kafka with KRaft

A Beginner's Journey to Simplifying Cluster Management

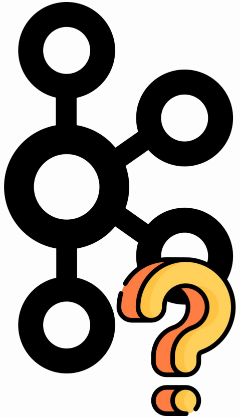
By **Nanthakumaran Senthilnathan**

About me

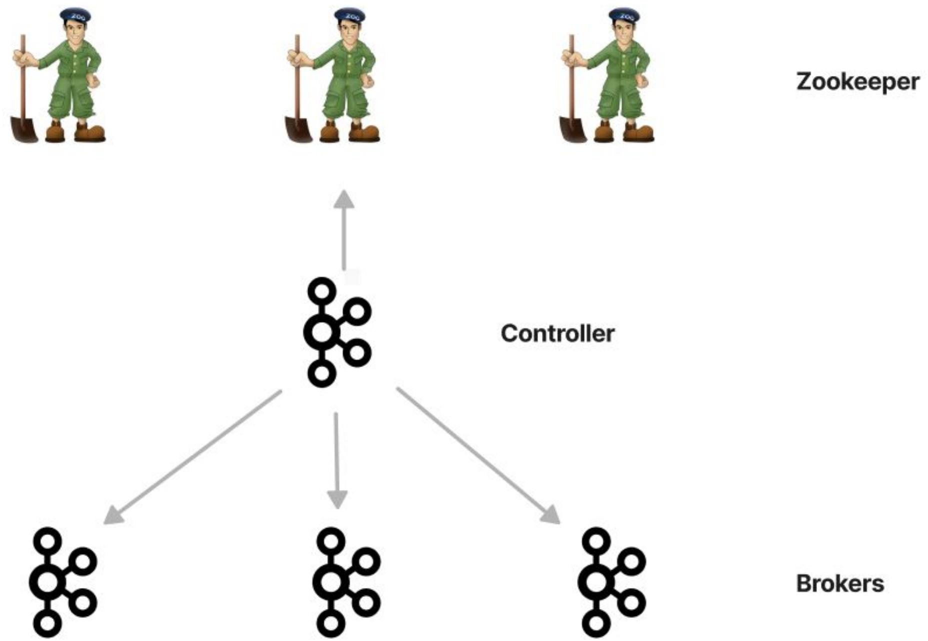
- Associate Engineer at Presidio
- AWS Certified Solutions Architect Associate & AWS Certified Developer Associate
- Regular speaker on Apache Kafka, distributed systems, and software development best practices
- Active open-source contributor
- Passionate about exploring new technologies and solving complex problems in the tech industry

What is Kafka?

- Distributed event streaming platform
- Real-time data pipelines, messaging, log aggregation, and stream processing
- Consumers, Producers, Log
- Traditional Architecture:
  - Zookeeper
  - Controller
  - Broker



Traditional Architecture



What wrong?

- Complexity and Operational Overhead
- Scalability Limitations
- Consistency Challenges
- Separate resource allocation for Zookeeper
- Limited Fault Tolerance

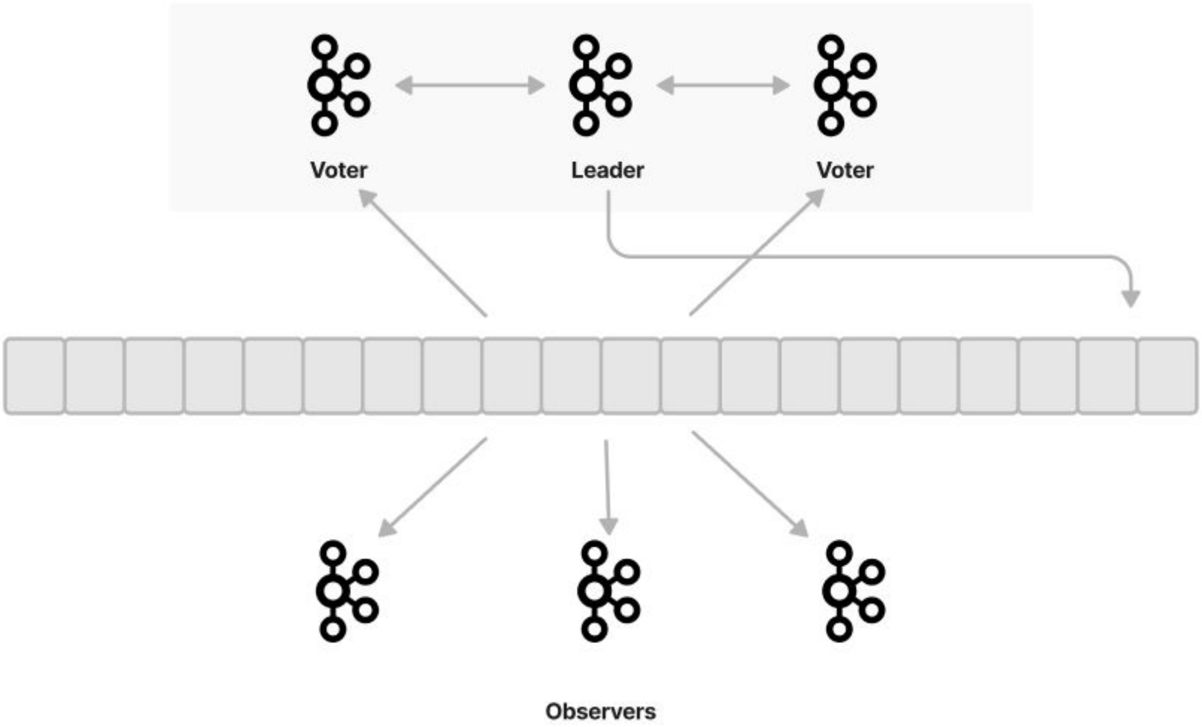
KRaft(King)’s Entry

- KRaft = Kafka Raft Metadata mode
- Eliminates Zookeeper dependency
- Part of Kafka Improvement Proposal (KIP-500)
- What have changed?
    - Simplified architecture
    - Improved scalability
    - Enhanced performance
    - Easier operations

Components of KRaft

- Raft consensus algorithm
- Controller quorum
- Metadata as internal topic

KRaft Architecture



KRaft vs Zookeeper

- Key differences:
    - Consensus protocol
    - Metadata storage
    - Scalability
- Advantages of KRaft:
    - Single system to manage
    - Improved fault tolerance
    - Better performance for metadata operations

Let’s see it in action!

System Benchmarks

Latest Progressions

- Kafka Tiered Storage (KIP-405)
- Kafka Streams DSL Improvements
- Enhanced Security: Finer-grained access controls
- Cluster-wide dynamic log adjustment for Kafka Connect
- Queues for Kafka

Thanks!

Do you need further clarification on anything?

nanthankumaransenthil@gmail.com
nanthakumaran.com

linkedin.com/in/nanthakumaran-s